Subgraph Federated Learning with Global Graph Reconstruction

Zhi Liu[1], Hanlin Zhou[1], Feng Xia[2], Guojiang Shen[1], Vidya Saikrishna[3], Xiaohua He[1], Jiaxin Du[1](✉), and Xiangjie Kong[1]

[1] Zhejiang University of Technology, Hangzhou 310014, China
jiaxin.joyce.du@gmail.com
[2] RMIT University, Melbourne, VIC 3000, Australia
[3] Federation University, Ballarat, VIC 3353, Australia

Abstract. Missing cross-subgraph information is a unique problem in Subgraph Federated Learning (SFL) and severely affects the performance of the learned model. Existing cutting-edge methods typically allow clients to exchange data with all other clients to predict missing neighbor nodes. However, such client-to-client data exchanges are highly complex and lead to expensive communication overhead. In this paper, we propose FedGGR: subgraph federated learning with global graph reconstruction. FedGGR is a practical and effective framework. Specifically, the core idea behind it is to directly learn a global graph on the server by a graph structure learning module instead of predicting the missing neighbors on each client. Compared to existing methods, FedGGR does not require any data exchange among clients and achieves remarkable enhancements in model performance. The experimental results on four benchmark datasets show that the proposed method excels with other state-of-the-art methods. We release our source code at https://github.com/poipoipoi233/FedGGR.

Keywords: Federated learning · Subgraph federated learning · Graph neural networks

1 Introduction

Graphs are prevalent to represent the interactions among entities in many real-world scenarios, such as social networks [1], traffic networks [2], and molecules [3]. To mine graph data, modern approaches generally resort to Graph Neural Networks (GNNs) [4]. Following a message-passing mechanism to aggregate each node's neighbor information and create node embedding for all nodes, GNNs can achieve outstanding performance in various graph mining tasks.

Most existing studies centrally train GNNs on a single complete graph, where node features and all connection information are gathered on a central server [5]. However, in certain real-world scenarios, each subgraph of such a complete graph may be stored at different data owners and only accessible locally. Consider an

X. Song et al. (Eds.): APWeb-WAIM 2023, LNCS 14331, pp. 158–173, 2024.
https://doi.org/10.1007/978-981-97-2303-4_11

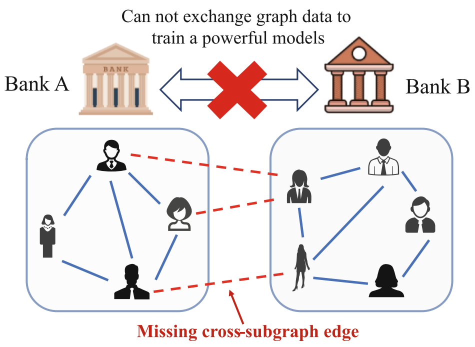

Can not exchange graph data to train a powerful models
Bank A
Bank B
Missing cross-subgraph edge

Fig. 1. A motivation scenario of subgraph federated learning. Two banks keep respective customer relation graphs for an identical graph mining task. However, they cannot share graph data to train a model collaboratively due to privacy concerns. The dashed red lines represent missing cross-subgraph edges. (Color figure online)

example in the financial domain, as shown in Figure 1. Two banks maintain their respective customer relation graphs, both aiming to address an identical task, such as fraud detection. However, due to privacy concerns or interest conflicts, each bank cannot share its graph data with others and merge them into a large graph for collaborative training. Therefore, such graph islands problem severely hinders the deployment of GNNs in practice.

Federated Learning (FL) [6] has experienced substantial progress in the past few years. It allows data owners, also known as clients, to train a global model collaboratively without sharing local data [7]. Employing FL to train GNNs in distributed subgraph scenarios, so-called Subgraph Federated learning (SFL), seems the simplest way to achieve collaborative training across different subgraph owners.

Nevertheless, training GNNs within federated subgraph scenarios introduces a unique challenge: *dealing with missing cross-subgraph neighbors and edges*. Specifically, nodes in each subgraph can potentially connect with nodes in other subgraphs, but no clients record such important cross-subgraph links. This issue is not trivial since these missing edges and potential neighbor nodes in other subgraphs may carry valuable information. Several recent studies have demonstrated that simply ignoring the missing cross-subgraph information will significantly impair the model accuracy of SFL [8,9].

Prior Work. Many cutting-edge works have made efforts to deal with the cross-subgraph information. For instance, FedSage+ [10] and FedNI [11] enable each client to predict cross-subgraph neighbors for local subgraph augmentation. Despite their proven effectiveness, these methods suffer from the drawback that clients must exchange large amounts of additional data (e.g., all node embeddings, model weights, and gradients) with all other clients. It incurs expensive communication overheads that grow exponentially with the number of clients.

Additionally, FedGCN [9] and FedGraph [8] assume that clients or server has detailed information about the original global graph structure. Therefore, in these methods, clients can acquire the node embeddings of their cross-subgraph neighbors directly based on the known global graph structure. Unfortunately, such assumptions are hard to be realized in reality. Moreover, these methods still require direct data exchange among clients.

Research Problem. Given the above realistic challenge, we propose the following question: *Can we provide a practical approach to learning the missing cross-subgraph information without client-to-client data exchange?*

Present Work. In this paper, we propose a subgraph federated learning framework named FedGGR. We designed it based on a simple and practical idea: directly learning the underlying global graph on the server side instead of predicting missing neighbor nodes on each client. Specifically, to realize this idea, we design and develop a global graph structure learning (GSL) module on the server to learn an optimal global graph structure and global node embeddings for all nodes. Each learned global node embedding would contain cross-subgraph knowledge and be sent to the corresponding client for local subgraph augmentation. Compared to existing works, this novel design avoids client-to-client data exchange and can fully explore potential cross-subgraph information to provide performance gain. However, the attendant challenge is that the standard FL framework does not support server-side model training. To further overcome this challenge, we draw inspiration from split learning [12] to design a training procedure. Concretely, FedGGR aggregates the gradient of the global node embeddings to update the server-side model and uses FedAvg to update the model on the client side.

We summarise our contributions in mainly three aspects:

- We propose a subgraph federated learning framework FedGGR, which can effectively reconstruct the missing cross-subgraph information. Moreover, it overcomes the limitations of relevant works that require client-to-client data exchange.
- We novelly propose a global GSL module on a server to learn the global graph directly. Besides, we further design a split learning-based training procedure to optimize the module. To the best of our knowledge, it is the first systematic attempt to address missing cross-subgraph information via graph structure learning.
- We conduct extensive experiments on four benchmark graph datasets. The results show that our FedGGR outperforms the state-of-the-art baselines.

2 Related Work

2.1 Subgraph Federated Learning (SFL)

An increasing number of studies on SFL have been published in recent years. For instance, FedSage+ [10] is a cornerstone study. In FedSage+, each client first requests additional data from other clients for subgraph augmentation. Then clients take the augmented subgraph to train a GNN model in the FL manner. Besides, FEDPUB [13] and FedEgo [14] focus on designing personalized FL strategies to address subgraph data heterogeneity among clients. Despite their effectiveness, these methods do not involve handling missing edges explicitly. As a result, performance degradation may inevitably occur. Another research direction is optimizing model convergence and communication overhead in the SFL setting. The representative works include FedGCN [9] and FedGraph [8]. However, these methods typically assume that clients or servers have information about all cross-subgraph edges and neighbors. Although the above works have provided valuable insights into SFL scenarios, reconstructing missing cross-subgraph information without exchanging data among clients remains a research gap. In this work, we aim to use our FedGGR to address this non-trivial challenge.

2.2 Graph Structure Learning (GSL)

The message-passing mechanism enables GNNs to take full advantage of the given graph structure information to learn the node embeddings. However, it also leads to GNNs being highly sensitive to the quality of the given graph structures [15]. This challenge motivates the development of graph structure learning [16], which aims to simultaneously learn an optimal graph structure and GNN parameters to obtain better performance on graph mining tasks. Certain studies, such as [17,18], have also incorporated GSL technologies into the federated setup. Nevertheless, no prior study has utilized GSL to address the issue of missing cross-subgraph information.

2.3 Split Learning

Split learning [12] is a distributed machine learning paradigm. Different from FL, the critical idea of split learning is to split a large neural network into multiple sub-networks, and each sub-network is trained on different parties [19]. In detail, the training of each sub-network is based on the exchange of the intermediate results (e.g., embedding) among participants. Several works have made progress toward combining split learning into FL. The application case includes federated spatiotemporal modeling [20] and federated news recommendation [21]. In this paper, we use an approach based on split learning to optimize the model parameter of our server-side global graph structure learning model.

3 Problem Setting

Consider $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$ as a global graph, where $\mathcal{V}$ is a set of $N = |\mathbf{V}|$ nodes, $\boldsymbol{X} \in \mathbb{R}^{N \times f}$ is the node feature matrix, and $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix that describes the pairwise relationship among nodes.

In SFL scenario involve m clients, each client $i \in [1, m]$ holds a small subgraph $\mathcal{G}_i = (\mathcal{V}_i, \boldsymbol{X}_i, \boldsymbol{A}_i)$ of the global graph $\mathcal{G}$ and all cross-subgraph edges present in the global graph are missing. A central server with sufficient computing power will coordinate clients for collaborative training. In this work, our goal is to design a practical SFL framework to deal with the performance degradation caused by missing cross-subgraph information. We assume that nodes of different subgraphs are disjoint and choose semi-supervised node classification as our downstream task, which is widely used in the performance evaluation of SFL frameworks.

4 Methodology

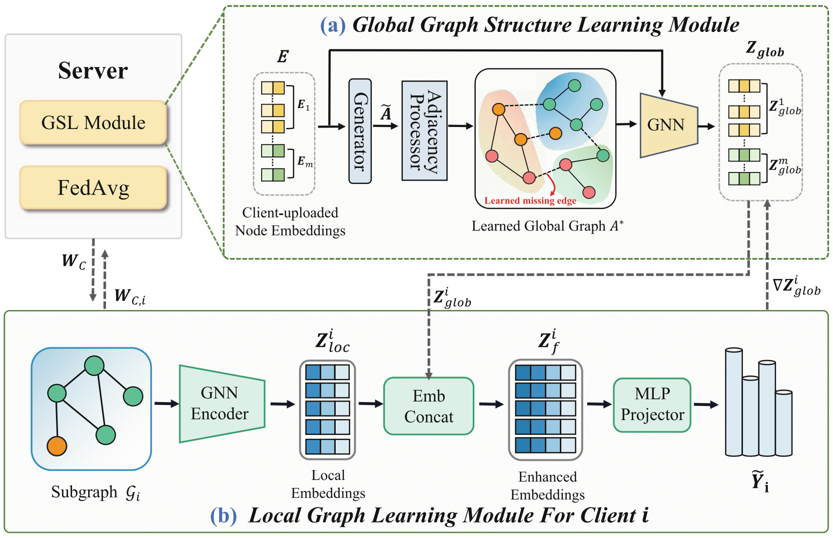


Fig. 2. Proposed FedGGR. Before federated training, clients upload compressed node embeddings to the server (described in Sect. 4.2). We present a detailed training pipeline in Sect. 4.5.

4.1 Framework Overview

We begin by giving a general overview of the framework. Figure 2 shows two core modules of our framework: the *global GSL module* on the server side and the *local graph learning module* on the client side.

More specifically, the global GSL module aims to directly learn the global graph structure and generate a global node embedding matrix $\boldsymbol{Z}_{glob}$, which incorporates cross-subgraph neighbor information. Then, the server will send each sub-matrix $\boldsymbol{Z}^{i}_{glob}$ to the corresponding client i as supplementary knowledge. In comparison, the local graph learning module aims to use a GNN model to learn the local node embeddings on their subgraph and then fuse the local node embeddings with the global node embeddings to perform downstream tasks.

Overall, our design has the following advantages. Firstly, directly learning the global graph on the server is an effective and easy-to-implement way to reconstruct the missing cross-subgraph information, which does not require clients to exchange any additional data with other clients. Secondly, fusing the global and local node embeddings will enhance each node's representation ability, which can significantly boost the performance of downstream tasks (demonstrated in Sect. 5.3).

4.2 Local Pre-training

Here, we first introduce a simple pre-training step before federated training. As previously noted, we aim to learn the global graph on the server directly. However, existing GSL technologies need the feature vector of each node as the starting point for training.

To address the above challenge, we propose to send each node's compressed embedding to the server. Technically, each client first trains a simple Graph Auto Encoder (GAE) [22] variant based on its local subgraph data. Subsequently, the trained GAE can compress each node's feature vector into a low-dimensional node embedding for uploading.

Take the client i as an example. Given a graph feature matrix $\boldsymbol{X}_i \in \mathbb{R}^{|\mathcal{V}_i|\times F}$, the forward process of our GAE is as follows:

$$\boldsymbol{E}_i = \text{Encoder}(\boldsymbol{X}_i, \boldsymbol{A}_i), \boldsymbol{X}_i^{rec} = \text{Decoder}(\boldsymbol{E}_i, \boldsymbol{A}_i), \tag{1}$$

where $\boldsymbol{E}_i \in \mathbb{R}^{|\mathcal{V}_i|\times p}$ denotes the compressed node embedding matrix. $\boldsymbol{X}_i^{rec} \in \mathbb{R}^{|\mathcal{V}_i|\times F}$ is the reconstructed node feature matrix. For simplicity, both Encoder and Decoder are two-layer GCN [23]. We optimize them by minimizing the mean squared error loss between $\boldsymbol{X}_i$ and $\boldsymbol{X}_i^{rec}$:

$$\mathcal{L}_i^{rec} = \|\boldsymbol{X}_i, \boldsymbol{X}_i^{rec}\|_F \tag{2}$$

After pre-training, each client i will use the encoder of trained GAE to generate $\boldsymbol{E}_i$ and upload it to the server.

4.3 The Local Graph Learning Module

Given a client i with subgraph $\mathcal{G}_i$, the local graph learning module first uses a GNN-based encoder f_θ to extract the local node embeddings:

$$\boldsymbol{Z}^i_{loc} = f_\theta(\mathcal{G}_i) = f_\theta(\boldsymbol{X}_i, \boldsymbol{A}_i), \tag{3}$$

where θ is the parameters of f_θ, $\boldsymbol{Z}^i_{loc} \in \mathbb{R}^{|\mathcal{V}_i| \times F_l}$ is the local node embedding matrix, and F_l is the dimension number. In FedGGR, we choose a two-layer GCN [23] as the encoder for a fair comparison with baseline methods.

To provide each node with knowledge about its missing cross-subgraph neighbor, we concatenate the local node embedding matrix $\boldsymbol{Z}^i_{loc}$ with the global node embedding matrix $\boldsymbol{Z}^i_{glob} \in \mathbb{R}^{|\mathcal{V}_i| \times F_g}$ generated from the global GSL module (described in Sect. 4.4). We denote the enhanced node embedding matrix as $\boldsymbol{Z}^i_f \in \mathbb{R}^{|\mathcal{V}_i| \times q}$, where $q = F_l + Fg$.

Finally, we use a one-layer MLP projector with the softmax function to obtain the final classification result:

$$\widetilde{\mathbf{Y}}_i = \text{SoftMax}(\text{MLP}(\boldsymbol{Z}^i_f)), \tag{4}$$

where the final output $\widetilde{\mathbf{Y}}_i \in \mathbb{R}^{|V_i| \times c}$ denotes the label prediction, and c denotes the number of classes. We use the cross-entropy loss function as the local loss function for each client i:

$$\mathcal{L}_i = \text{CrossEntropyLoss}(\widetilde{\mathbf{Y}}_i, \mathbf{Y}_i), \tag{5}$$

where Y_i denotes the set of labeled nodes on the subgraph $\mathcal{G}_i$.

4.4 The Global Graph Structure Learning Module

The global GSL module, as shown in Fig. 2 (a), consists of three components: a graph generator, an adjacency processor, and a GNN model.

Graph Generator. It takes the embedding matrix $\boldsymbol{E} = \text{CONCAT}(\boldsymbol{E}_1, \cdots, \boldsymbol{E}_m)$ as input, where $\boldsymbol{E}_i$ is generated by the local pre-training step. The output of the graph generator is an initial global adjacency matrix $\widetilde{\boldsymbol{A}}$. We choose the MLP-D generator proposed by [16] as the graph generator in our FedGGR. Formally, the MLP-D generator is defined as:

$$\widetilde{\boldsymbol{A}} = \text{KNN}(\text{MLP}(\boldsymbol{E})), \tag{6}$$

where KNN is the K nearest neighbor algorithm, MLP is a two-layer MLP projector. The weight of this MLP is zero, except for the primary diagonal.

Adjacency Processor. Following [16], we use the adjacency processor to make the initial global adjacency matrix normalized and symmetric. Specifically, the adjacency processor performs the following transformation on the adjacency matrix $\widetilde{\boldsymbol{A}}$ to obtain a refined global adjacency matrix $\boldsymbol{A}^*$:

$$\boldsymbol{A}^* = \frac{1}{2}\boldsymbol{D}^{-\frac{1}{2}}(P(\widetilde{\boldsymbol{A}}) + P(\widetilde{\boldsymbol{A}})^T)\boldsymbol{D}^{-\frac{1}{2}}, \tag{7}$$

where $\boldsymbol{D}$ is the degree matrix of $\widetilde{\boldsymbol{A}}$, and P is the ELU [24] activation function.

Global GNN Model. The inputs of the global GNN model are the refined global adjacency matrix $\boldsymbol{A}^*$ and the node embedding matrix $\boldsymbol{E}$. It outputs the corresponding global embedding for each node in the global graph. The forward propagation process of the global GNN model is as follows:

$$\mathbf{Z}_{glob} = f_\varphi(\boldsymbol{E}, \mathbf{A}^*), \tag{8}$$

where $\boldsymbol{Z}_{glob} \in \mathbb{R}^{|\mathcal{V}| \times F_g}$ is the global node embedding matrix, F_g is its dimension number. $f_\varphi(\cdot)$ is a GNN encoder with the parameter φ. We also choose a two-layer GCN as the GNN encoder here. In $\boldsymbol{Z}_{glob}$, the submatrix $\boldsymbol{Z}^i_{glob}$ contains the global node embeddings corresponding to the nodes of the client i. Each global node embedding has already integrated cross-subgraph information thanks to the learned global adjacency matrix and the message-passing mechanism of GNNs. The server will send each $\mathbf{Z}^i_{glob}$ to the corresponding client i.

4.5 Objective and Training Procedure

In this subsection, we state our training objective: We look for a set of optimal parameters (W^*_C, W^*_S) that can minimize the classification losses of all clients.

$$(W^*_C, W^*_S) = \underset{(W_C, W_S)}{\arg\min} \frac{1}{M} \sum_{i=1}^{M} \frac{|\mathcal{V}_i|}{|\mathcal{V}|} \mathcal{L}_i(\mathcal{G}_i; W_C; W_S), \tag{9}$$

where $\mathcal{L}_i$ represents the local loss of client i. W_C and W_S denote the learnable parameters in the local graph learning and global GSL modules, respectively.

The training procedure consists of two parts: (1) clients pre-train a GAE model. Subsequently, they use the trained GAE to generate compressed node embeddings, which they then upload to the server. (2) In each communication round, FedGGR employs the FedAvg [6] algorithm to update the W_C collaboratively, and the server updates W_S by aggregating the gradients $\nabla \boldsymbol{Z}_{glob}$ of the global node embeddings $\boldsymbol{Z}_{glob}$. We present a detailed training workflow in Algorithm 1.

Algorithm 1. The training algorithm of FedGGR

```
Input: $\mathcal{G}_i$, $i = 1, \cdots, m$, learning rate $\eta$, number of nearest neighbors $k$
Output: Trained $W_S$ and $W_C$
Client Pre-training: each client $i$ pre-trains a GAE model and then uploads the compressed node embedding $E_i$ to the server.
Server Executes:
1: Initialize:$W_S^{(0)}$ and $W_C^{(0)}$;
2: $\boldsymbol{E} \leftarrow$ CONCAT($\boldsymbol{E}_1, \cdots, \boldsymbol{E}_M$);
3: for each round $t = 1, 2, \cdots$ do
4:     $\boldsymbol{Z}_{glob} \leftarrow$ GlobalGSL($\boldsymbol{E}, k$) // Eq. (6), (7), and (8);
5:     for each client $i$ in parallel do
6:         $W_{C,i}^{(t)}, \nabla \boldsymbol{Z}_{glob}^i \leftarrow$ ClientTraining($i, W_C^{(t)}, \boldsymbol{Z}_{glob}^i$)
7:     end for
8:     $\nabla \boldsymbol{Z}_{glob} \leftarrow$ CONCAT($\nabla \boldsymbol{Z}_{glob}^1, \cdots, \nabla \boldsymbol{Z}_{glob}^m$);
9:     $\nabla W_S^{(t)} \leftarrow \boldsymbol{Z}_{glob}.backward(\nabla \mathbf{Z}_{glob})$;
10:    $W_C^{(t+1)} \leftarrow \sum_i \frac{|V_i|}{|V|} W_{C,i}^{(t)}$;
11:    $W_S^{(t+1)} \leftarrow W_S^{(t)} - \eta \nabla W_S^{(t)}$;
12: end for
ClientTraining($i, W_C^{(t)}, \mathbf{Z}_{glob}^i$) :
1: $W_{C,i}^{(t)} \leftarrow W_C^{(t)}$;
2: for each local epoch $j = 1, \cdots, Q$ do
3:     $\mathcal{L}_i \leftarrow$ Eq. (5) with $Z_{glob}^i$
4:     $\nabla W_{C,i}^{(t)}, \nabla \mathbf{Z}_{glob}^i \leftarrow \mathcal{L}_i.backward()$
5:     $W_{C,i}^{(t)} \leftarrow W_{C,i}^{(t)} - \eta \nabla W_{C,i}^{(t)}$
6: end for
7: return $W_{C,i}^{(t)}$ and $\nabla \mathbf{Z}_{glob}^i$ to server
```

5 Experiment

In this section, we conduct experiments to verify our proposed FedGGR. We run all experiments on FederatedScope-GNN (FS-G) [25], a comprehensive Python package for federated graph learning. We aim to answer three questions:

1. **RQ1:** How does FedGGR perform compared with the state-of-the-art SFL baseline methods?
2. **RQ2:** Are the learned global graph and the global node embeddings really useful?
3. **RQ3:** How do critical hyper-parameters impact the performance of FedGGR?

5.1 Experimental Setups

Datasets and Graph Partitioning. We test FedGGR on four benchmark datasets, i.e., Cora [26], CiteSeer [26], PubMed [27], and Amazon-Computers [28]. The dataset statistics are shown in Table 1. We split the complete global graph into 3, 5, and 10 subgraphs for all datasets using the *random_splitter*, a graph partitioning algorithm provided by FS-G [25]. Specifically, the *random_splitter* will randomly split the set of nodes in the original graph into m subsets and then deduce subgraphs based on the node subsets. It should be noted that the *random_splitter* will produce more missing cross-subgraph edges and decrease the homophilic degree in each subgraph compared to the community detection-based graph partitioning algorithm (e.g., Louvain [29]). We also specify that the nodes of each subgraph are non-overlapping since it will produce the highest number of missing edges.

Baselines. We make comparisons with various baselines to demonstrate the performance of our FedGGR. The chosen baselines are as follows:

1. **Typical GNNs**: We use the FedAvg [6] to train several representative GNNs, i.e., GCN [30], GAT [31], and GarphSAGE [32] in the SFL setting.
2. **FedSage+** [10]: This method is the most representative SFL baseline. The basic idea of this method is to generate the missing neighbors on each client with the help of exchanging additional data among clients.
3. **GCLF+** [33]: This baseline is a well-known personalized FL method designed for the graph-level FL setting. In GCFL+, the server clusters the clients according to the gradient sequence of each client's local model, and the server will generate multiple global models for each client cluster. We slightly adapt this method to make it applicable to our SFL setting.
4. **FED-PUB** [13]: It is a state-of-the-art SFL baseline. In this method, each client's local GNN takes a random graph as input to compute the functional embeddings. Then, each client uploads the functional embeddings to the server. The server then calculates the subgraph similarity matrix according to the function embeddings and generates customized models for each client based on the similarity matrix.

Table 1. Graph Data Statistics

Dataset	Node	Edge	Feature	Classes
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3
Computers	13,752	491,722	767	10

Implementation Details. Following [25,34], we divide the set of nodes into train/valid/test sets, with a ratio of 60:20:20. We search the optimal number of hidden units for GCN within $\{64, 128, 256\}$. The F_l and F_g are selected in the range of $\{64, 128, 256\}$. For the graph generator in the global GSL module, we vary the number of neighbors k over $\{5, 10, ..., 40\}$. For training, we use the Adam optimizer to optimize all model parameters. We vary the learning rate

$\eta \in \{1e^{-3}, 1e^{-2}\}$ and the number of local epochs $Q \in \{1, 2, 3\}$. The initial hyper-parameter settings of baseline methods are based on the original papers and official public code. We further perform an optimal hyper-parameter search for all baseline methods to make a fair comparison.

Table 2. Comparison results on representative node classification datasets with *random splitter*. The reported results are mean accuracy with standard deviation on the test sets over five different runs.

Method	Cora			CiteSeer		
	3 Clients	5 Clients	10 Clients	3 Clients	5 Clients	10 Clients
GCN	82.3 0.5	80.5 0.8	78.7 0.6	76.6 0.5	75.4 0.6	76.3 0.8
GAT	82.6 0.5	80.1 0.3	78.5 0.6	76.8 0.5	75.7 0.5	76.0 0.8
GraphSAGE	83.2 0.5	80.2 0.4	78.8 0.7	76.1 0.2	74.9 0.3	73.8 0.2
GCFL+	83.4 0.6	80.1 0.4	78.4 1.3	77.4 0.6	76.1 0.1	73.9 0.6
FedSage+	83.7 0.6	81.1 0.2	77.2 0.2	77.3 0.4	75.6 0.5	75.4 0.8
FED-PUB	83.9 0.6	81.1 1.1	76.2 0.8	76.9 0.7	76.7 0.4	76.7 0.4
FedGGR(ours)	84.1 0.6	81.8 1.1	79.5 0.6	79.0 0.9	77.6 0.9	77.0 0.7
Method	PubMed			Amazon-Computers		
	3 Clients	5 Clients	10 Clients	3 Clients	5 Clients	10 Clients
GCN	85.6 0.6	84.8 1.0	85.4 1.2	88.4 0.9	87.6 0.8	83.8 0.2
GAT	84.6 0.6	85.0 1.1	84.4 0.1	87.9 0.5	86.6 0.5	85.3 0.7
GraphSAGE	85.8 1.6	85.7 0.6	88.6 0.1	85.1 0.3	84.0 0.3	83.7 0.8
GCFL+	85.5 0.3	85.9 0.5	86.3 0.1	85.8 0.4	85.2 0.1	84.9 0.1
FedSage+	87.4 0.4	86.0 1.5	86.2 1.1	87.2 0.5	86.7 0.3	85.1 0.1
FED-PUB	89.7 0.3	89.4 0.1	88.8 0.3	89.1 0.6	88.9 0.8	86.2 0.8
FedGGR(ours)	90.0 0.1	89.5 0.2	89.1 0.4	89.8 0.1	89.0 0.2	86.8 0.2

5.2 Comparison with State-of-the-art Methods (RQ1)

Table 2 presents the node classification performance of our FedGGR and baseline methods. Overall, we have the following observations.

Firstly, our FedGGR consistently outperforms all baselines on four benchmark datasets. Notably, FedGGR does not employ additional strategies to address the data heterogeneity for simplicity. However, it can be easily integrated with existing personalized federated learning methods to improve performance further.

Secondly, the performance of all methods decreases with the increase in the number of clients on most datasets. This phenomenon is reasonable since the number of missing edges and the heterogeneity between subgraphs increase with the number of clients.

Finally, we observe that the performance of some methods does not decrease but fluctuates as the number of clients increases on the PubMed dataset. This unusual phenomenon may be because some unnecessary noisy edges are removed during the subgraph partitioning process.

5.3 Ablation Study (RQ2)

We do an ablation study on the global GSL module in this subsection. In fact, the GCN trained by FedAvg can be seen as a variant of FedGGR without the global GSL module. As shown in Table 2, FedGGR performs better than the GCN by an average of 2.7% on the four datasets. This observation demonstrates that the FedGGR significantly benefits from the global GSL module.

For further demonstration, we build variants of FedGGR by replacing the GNN encoders of the local graph learning module with other representative GNNs, i.e., GAT [31] and GPR-GNN [34]. Figure 3 presents the test accuracy of these variants on the CiteSeer dataset with and without the global GSL module. It shows that the global GSL module can provide considerable performance gains regardless of the backbone model.

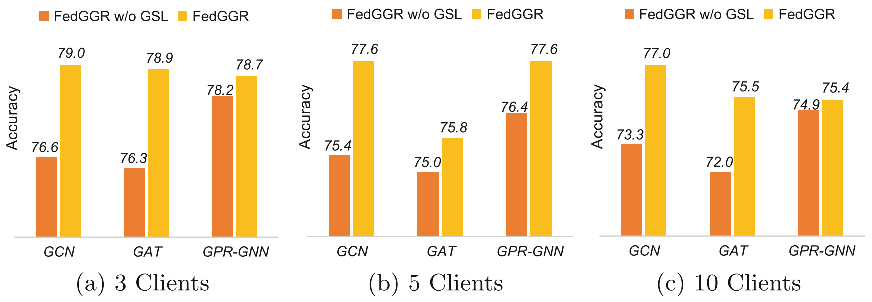
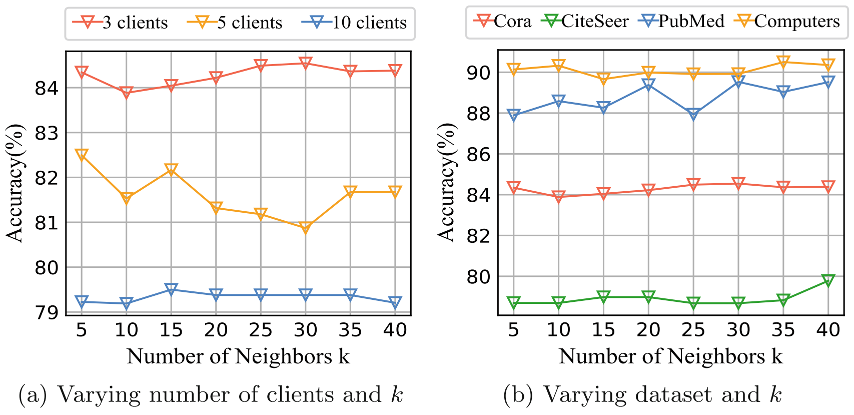


Fig. 3. Node classification accuracy of FedGGR and its variant on the CiteSeer dataset.

5.4 Sensitivity Analysis (RQ3)

In this subsection, we analyze the sensitivity of critical hyper-parameters in our FedGGR. Specifically, the analyzed hyper-parameters include the number of neighbors k in KNN post-processing of the global GSL module, the dimension number of the global and local embeddings F_l and F_g, and the number of local update epochs Q.

Number of Neighbors *k*. This hyperparameter determines the number of neighbors for each node when generating the global graph structure. We change the value of k in the $\{5, 10, ..., 40\}$ under different datasets and varied numbers of clients. We have two main observations. Firstly, as shown in Fig. 4(a), the best choice of k is different under different numbers of clients. Secondly, Fig. 4(b) shows that the change of k has an insignificant effect on the final test accuracy. This observation differs from the typical results observed in centralized graph structure learning works, where an unsuitable k will significantly reduce the accuracy. For instance, an inappropriate k value causes a drop in accuracy of more than 5% for the Cora dataset in work [16]. A explanation for this observation is that the local node embeddings in FedGGR mitigate the harmful effects of bad global embeddings generated by setting an inappropriate k.



(a) Varying number of clients and k (b) Varying dataset and k

Fig. 4. The node classification accuray about k under different settings.

Local Update Epochs Q. We compare the node classification performance under different Q values with three clients. Table 3 shows that the best Q value is one for the three citation network datasets, i.e., Cora, CiteSeer, and PubMed. While for the Computer dataset, setting the Q value to three produces the highest accuracy. Notably, increasing the number of local updates is more likely to lead to overfitting, especially when the number of nodes in each subgraph is small. We recommend that readers tune this hyperparameter according to the dataset size for better performance.

Table 3. Comparison results by varying the number of local update epochs.

Datasets	Cora	CiteSeer	PubMed	Computers
$Q = 1$	**0.841**	**0.790**	**0.900**	0.863
$Q = 2$	0.827	0.788	0.893	0.890
$Q = 3$	0.826	0.789	0.887	**0.898**

The Output Dimension F_l and F_g. We investigate the impact of different combinations of F_l and F_g on accuracy. Figure 5 shows the results on the PubMed dataset. We see that the accuracy is relatively stable when changing these two hyper-parameters, demonstrating the robustness of our FedGGR. Compared to F_l, FedGGR is less sensitive to the selection of F_g. In addition, we observe that the performance tends to be higher when the F_l is equal to F_g.

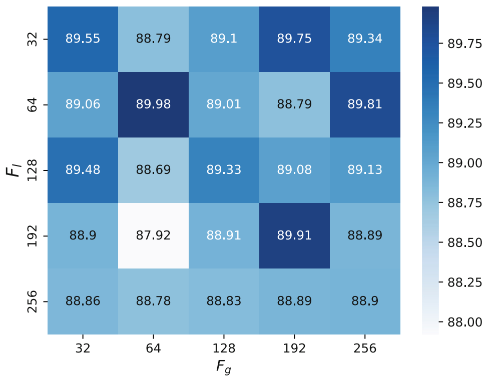


Fig. 5. Accuracy w.r.t. combinations of F_l and F_g on the PubMed dataset with three clients.

6 Conclusion

In this paper, we focus on dealing with missing cross-subgraph information in subgraph federated learning. We propose FedGGR, a quite practical subgraph federated learning framework. It enables the server to directly learn the global graph using a graph structure learning module and a split learning-based training procedure. Compared to the relevant methods, FedGGR not only eliminates the need for direct data exchange among clients but also offers remarkable performance gains in model accuracy. Extensive experiments on the four benchmark datasets validate the effectiveness of our proposed method.

References

1. Kong, X., Gao, H., Shen, G., Duan, G., Das, S.K.: FedVCP: a federated-learning-based cooperative positioning scheme for social internet of vehicles. IEEE Trans. Comput. Soc. Syst. **9**(1), 197–206 (2021)
2. Hou, M., Xia, F., Gao, H., Chen, X., Chen, H.: Urban region profiling with spatio-temporal graph neural networks. IEEE Trans. Comput. Soc. Syst. **9**(6), 1736–1747 (2022)
3. Xia, J., et al.: Mole-BERT: rethinking pre-training graph neural networks for molecules. In: The Eleventh International Conference on Learning Representations, pp. 1–18 (2023)
4. Zhou, J., et al.: Graph neural networks: a review of methods and applications. AI Open **1**, 57–81 (2020)
5. Xia, F., et al.: CenGCN: centralized convolutional networks with vertex imbalance for scale-free graphs. IEEE Trans. Knowl. Data Eng. **35**(5), 4555–4569 (2022)
6. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
7. Kong, X., et al.: A federated learning-based license plate recognition scheme for 5G-enabled internet of vehicles. IEEE Trans. Ind. Inf. **17**(12), 8523–8530 (2021)

8. Chen, F., Li, P., Miyazaki, T., Wu, C.: Fedgraph: federated graph learning with intelligent sampling. IEEE Trans. Parallel Distrib. Syst. **33**(8), 1775–1786 (2021)
9. Yao, Y., Joe-Wong, C.: FedGCN: convergence and communication tradeoffs in federated training of graph convolutional networks, pp. 1–31 (2022). arXiv preprint arXiv:2201.12433
10. Zhang, K., Yang, C., Li, X., Sun, L., Yiu, S.M.: Subgraph federated learning with missing neighbor generation. Adv. Neural. Inf. Process. Syst. **34**, 6671–6682 (2021)
11. Peng, L., Wang, N., Dvornek, N., Zhu, X., Li, X.: FedNI: federated graph learning with network inpainting for population-based disease prediction. IEEE Trans. Med. Imaging 1–12 (2022)
12. Gupta, O., Raskar, R.: Distributed learning of deep neural network over multiple agents. J. Netw. Comput. Appl. **116**, 1–8 (2018)
13. Baek, J., Jeong, W., Jin, J., Yoon, J., Hwang, S.J.: Personalized subgraph federated learning, pp. 1–20 (2022). arXiv preprint arXiv:2206.10206
14. Zhang, T., Chen, C., Chang, Y., Shu, L., Zheng, Z.: FedEgo: privacy-preserving personalized federated graph learning with ego-graphs, pp. 1–25 (2022). arXiv preprint arXiv:2208.13685
15. Liu, Y., Zheng, Y., Zhang, D., Chen, H., Peng, H., Pan, S.: Towards unsupervised deep graph structure learning. In: Proceedings of the ACM Web Conference 2022, pp. 1392–1403 (2022)
16. Fatemi, B., El Asri, L., Kazemi, S.M.: SLAPS: self-supervision improves structure learning for graph neural networks. Adv. Neural. Inf. Process. Syst. **34**, 22667–22681 (2021)
17. Chen, F., Long, G., Wu, Z., Zhou, T., Jiang, J.: Personalized federated learning with a graph. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, pp. 2575–2582 (2022)
18. Zhao, G., Huang, Y., Tsai, C.H.: FedGSL: federated graph structure learning for local subgraph augmentation. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 818–824. IEEE (2022)
19. Thapa, C., Arachchige, P.C.M., Camtepe, S., Sun, L.: SplitFed: when federated learning meets split learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 8485–8493 (2022)
20. Meng, C., Rambhatla, S., Liu, Y.: Cross-node federated graph neural network for spatio-temporal data modeling. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1202–1211 (2021)
21. Yi, J., Wu, F., Wu, C., Liu, R., Sun, G., Xie, X.: Efficient-FedRec: efficient federated learning framework for privacy-preserving news recommendation. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 2814–2824 (2021)
22. Kipf, T.N., Welling, M.: Variational graph auto-encoders, pp. 1–3 (2016). arXiv preprint arXiv:1611.07308
23. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, pp. 1–14 (2017)
24. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs). In: 4th International Conference on Learning Representations, pp. 1–14 (2016)
25. Xie, Y., et al.: FederatedScope: a flexible federated learning platform for heterogeneity. Proc. VLDB Endow. **16**(5), 1059–1072 (2023)
26. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–106 (2008)

27. Namata, G., London, B., Getoor, L., Huang, B., Edu, U.: Query-driven active surveying for collective classification. In: 10th International Workshop on Mining and Learning with Graphs, vol. 8, pp. 1–8 (2012)
28. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation, pp. 1–11 (2018). arXiv preprint arXiv:1811.05868
29. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech: Theory Exp. **2008**(10), P10008 (2008)
30. Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., Pei, J.: AM-GCN: adaptive multi-channel graph convolutional networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1243–1253 (2020)
31. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, pp. 1–12 (2018)
32. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Adv. Neural. Inf. Process. Syst. **30**, 1–11 (2017)
33. Xie, H., Ma, J., Xiong, L., Yang, C.: Federated graph classification over MNon-IID graphs. Adv. Neural. Inf. Process. Syst. **34**, 18839–18852 (2021)
34. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. In: 9th International Conference on Learning Representations, pp. 1–24 (2021)